

# libGIS

## 1.0.4

Generated by Doxygen 1.6.3

Thu Jul 8 20:50:57 2010



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	_AtmelGenericRecord Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Field Documentation . . . . .	5
3.1.2.1	address . . . . .	5
3.1.2.2	data . . . . .	5
3.2	_IHexRecord Struct Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.2.2	Field Documentation . . . . .	6
3.2.2.1	address . . . . .	6
3.2.2.2	checksum . . . . .	6
3.2.2.3	data . . . . .	6
3.2.2.4	dataLen . . . . .	6
3.2.2.5	type . . . . .	6
3.3	_SRecord Struct Reference . . . . .	7
3.3.1	Detailed Description . . . . .	7
3.3.2	Field Documentation . . . . .	7
3.3.2.1	address . . . . .	7
3.3.2.2	checksum . . . . .	7
3.3.2.3	data . . . . .	7
3.3.2.4	dataLen . . . . .	7
3.3.2.5	type . . . . .	7

<b>4</b>	<b>File Documentation</b>	<b>9</b>
4.1	atmel_generic.h File Reference	9
4.1.1	Detailed Description	10
4.1.2	Typedef Documentation	10
4.1.2.1	AtmelGenericRecord	10
4.1.3	Enumeration Type Documentation	10
4.1.3.1	AtmelGenericErrors	10
4.1.4	Function Documentation	11
4.1.4.1	New_AtmelGenericRecord	11
4.1.4.2	Print_AtmelGenericRecord	11
4.1.4.3	Read_AtmelGenericRecord	11
4.1.4.4	Write_AtmelGenericRecord	12
4.2	ihex.h File Reference	13
4.2.1	Detailed Description	14
4.2.2	Typedef Documentation	14
4.2.2.1	IHexRecord	14
4.2.3	Enumeration Type Documentation	14
4.2.3.1	IHexErrors	14
4.2.3.2	IHexRecordTypes	14
4.2.4	Function Documentation	15
4.2.4.1	Checksum_IHexRecord	15
4.2.4.2	New_IHexRecord	15
4.2.4.3	Print_IHexRecord	15
4.2.4.4	Read_IHexRecord	16
4.2.4.5	Write_IHexRecord	16
4.3	srecord.h File Reference	17
4.3.1	Detailed Description	18
4.3.2	Typedef Documentation	18
4.3.2.1	SRecord	18
4.3.3	Enumeration Type Documentation	18
4.3.3.1	SRecordErrors	18
4.3.3.2	SRecordTypes	18
4.3.4	Function Documentation	19
4.3.4.1	Checksum_SRecord	19
4.3.4.2	New_SRecord	19
4.3.4.3	Print_SRecord	20

---

4.3.4.4	Read_SRecord . . . . .	20
4.3.4.5	Write_SRecord . . . . .	20



# Chapter 1

# Data Structure Index

## 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">_AtmelGenericRecord</a>	5
<a href="#">_IHexRecord</a>	6
<a href="#">_SRecord</a>	7





# Chapter 2

## File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">atmel_generic.h</a> (Low-level utility functions to create, read, write, and print Atmel Generic binary records ) . . . . .	9
<a href="#">ihex.h</a> (Low-level utility functions to create, read, write, and print Intel HEX8 binary records ) .	13
<a href="#">srecord.h</a> (Low-level utility functions to create, read, write, and print Motorola S-Record binary records ) . . . . .	17



## Chapter 3

# Data Structure Documentation

### 3.1 `_AtmelGenericRecord` Struct Reference

```
#include <atmel_generic.h>
```

#### Data Fields

- `uint32_t` [address](#)
- `uint16_t` [data](#)

#### 3.1.1 Detailed Description

Structure to hold the fields of an Atmel Generic record.

#### 3.1.2 Field Documentation

##### 3.1.2.1 `uint32_t _AtmelGenericRecord::address`

The 24-bit address field of the record. Note that the Atmel Generic record only supports 24-bit addresses, despite the additional storage capacity (32 bits) of this variable.

##### 3.1.2.2 `uint16_t _AtmelGenericRecord::data`

The 16-bit data field of the record.

The documentation for this struct was generated from the following file:

- [atmel\\_generic.h](#)

## 3.2 `_IHexRecord` Struct Reference

```
#include <ihex.h>
```

### Data Fields

- `uint16_t address`
- `uint8_t data [IHEX_MAX_DATA_LEN/2]`
- `int dataLen`
- `int type`
- `uint8_t checksum`

### 3.2.1 Detailed Description

Structure to hold the fields of an Intel HEX8 record.

### 3.2.2 Field Documentation

#### 3.2.2.1 `uint16_t _IHexRecord::address`

The 16-bit address field.

#### 3.2.2.2 `uint8_t _IHexRecord::checksum`

The checksum of this record.

#### 3.2.2.3 `uint8_t _IHexRecord::data[IHEX_MAX_DATA_LEN/2]`

The 8-bit array data field, which has a maximum size of 256 bytes.

#### 3.2.2.4 `int _IHexRecord::dataLen`

The number of bytes of data stored in this record.

#### 3.2.2.5 `int _IHexRecord::type`

The Intel Hex record type of this record.

The documentation for this struct was generated from the following file:

- [ihex.h](#)

## 3.3 \_SRecord Struct Reference

```
#include <srecord.h>
```

### Data Fields

- `uint32_t address`
- `uint8_t data [SRECORD_MAX_DATA_LEN/2]`
- `int dataLen`
- `int type`
- `uint8_t checksum`

#### 3.3.1 Detailed Description

Structure to hold the fields of a Motorola S-Record record.

#### 3.3.2 Field Documentation

##### 3.3.2.1 `uint32_t _SRecord::address`

The address field. This can be 16-,24-, or 32-bits depending on the record type.

##### 3.3.2.2 `uint8_t _SRecord::checksum`

The checksum of this record.

##### 3.3.2.3 `uint8_t _SRecord::data[SRECORD_MAX_DATA_LEN/2]`

The 8-bit array data field, which has a maximum size of 32 bytes.

##### 3.3.2.4 `int _SRecord::dataLen`

The number of bytes of data stored in this record.

##### 3.3.2.5 `int _SRecord::type`

The Motorola S-Record type of this record (S0-S9).

The documentation for this struct was generated from the following file:

- [srecord.h](#)



# Chapter 4

## File Documentation

### 4.1 `atmel_generic.h` File Reference

Low-level utility functions to create, read, write, and print Atmel Generic binary records.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
```

#### Data Structures

- struct [\\_AtmelGenericRecord](#)

#### Typedefs

- typedef struct [\\_AtmelGenericRecord](#) [AtmelGenericRecord](#)

#### Enumerations

- enum [\\_AtmelGenericDefinitions](#) {  
    **ATMEL\_GENERIC\_RECORD\_BUFF\_SIZE** = 16, **ATMEL\_GENERIC\_ADDRESS\_LEN** =  
    6, **ATMEL\_GENERIC\_DATA\_LEN** = 4, **ATMEL\_GENERIC\_SEPARATOR\_OFFSET** = 6,  
    **ATMEL\_GENERIC\_SEPARATOR** = ':' }  
• enum [AtmelGenericErrors](#) {  
    **ATMEL\_GENERIC\_OK** = 0, **ATMEL\_GENERIC\_ERROR\_FILE** = -1, **ATMEL\_GENERIC\_-**  
    **ERROR\_EOF** = -2, **ATMEL\_GENERIC\_ERROR\_INVALID\_RECORD** = -3,  
    **ATMEL\_GENERIC\_ERROR\_INVALID\_ARGUMENTS** = -4, **ATMEL\_GENERIC\_ERROR\_-**  
    **NEWLINE** = -5 }

## Functions

- int [New\\_AtmelGenericRecord](#) (uint32\_t address, uint16\_t data, [AtmelGenericRecord](#) \*genericRecord)
- int [Read\\_AtmelGenericRecord](#) ([AtmelGenericRecord](#) \*genericRecord, FILE \*in)
- int [Write\\_AtmelGenericRecord](#) (const [AtmelGenericRecord](#) genericRecord, FILE \*out)
- void [Print\\_AtmelGenericRecord](#) (const [AtmelGenericRecord](#) genericRecord)

### 4.1.1 Detailed Description

Low-level utility functions to create, read, write, and print Atmel Generic binary records.

#### Author

Vanya A. Sergeev <[vsergeev@gmail.com](mailto:vsergeev@gmail.com)>

#### Date

October 10 2009

#### Version

1.0.3

### 4.1.2 Typedef Documentation

#### 4.1.2.1 typedef struct [\\_AtmelGenericRecord](#) [AtmelGenericRecord](#)

Alias "[AtmelGenericRecord](#)" for struct [\\_AtmelGenericRecord](#), done for convenience and clarity.

### 4.1.3 Enumeration Type Documentation

#### 4.1.3.1 enum [AtmelGenericErrors](#)

All possible error codes the Atmel Generic record utility functions may return.

#### Enumerator:

***ATMEL\_GENERIC\_OK*** Error code for success or no error.

***ATMEL\_GENERIC\_ERROR\_FILE*** Error code for error while reading from or writing to a file.  
You may check errno for the exact error if this error code is encountered.

***ATMEL\_GENERIC\_ERROR\_EOF*** Error code for encountering end-of-file when reading from a file.

***ATMEL\_GENERIC\_ERROR\_INVALID\_RECORD*** Error code for error if an invalid record was read.

***ATMEL\_GENERIC\_ERROR\_INVALID\_ARGUMENTS*** Error code for error from invalid arguments passed to function.

***ATMEL\_GENERIC\_ERROR\_NEWLINE*** Error code for encountering a newline with no record when reading from a file.



### 4.1.4 Function Documentation

#### 4.1.4.1 `int New_AtmelGenericRecord (uint32_t address, uint16_t data, AtmelGenericRecord * genericRecord)`

Sets all of the record fields of an Atmel Generic record structure. Note that the Atmel Generic record only supports 24-bit addresses.

##### Parameters

*address* The 24-bit address of the data.

*data* The 16-bit word of data.

*genericRecord* A pointer to the target Atmel Generic record structure where these fields will be set.

##### Returns

ATMEL\_GENERIC\_OK on success, otherwise one of the ATMEL\_GENERIC\_ERROR\_ error codes.

##### Return values

*ATMEL\_GENERIC\_OK* on success.

*ATMEL\_GENERIC\_ERROR\_INVALID\_ARGUMENTS* if genericRecord does not point to a valid AtmelGenericRecord structure.

#### 4.1.4.2 `void Print_AtmelGenericRecord (const AtmelGenericRecord genericRecord)`

Prints the contents of an Atmel Generic record structure to stdout. The record dump consists of the address and data fields of the record.

##### Parameters

*genericRecord* The Atmel Generic record structure that will be printed to stdout.

##### Returns

Always returns ATMEL\_GENERIC\_OK (success).

##### Return values

*ATMEL\_GENERIC\_OK* on success.

#### 4.1.4.3 `int Read_AtmelGenericRecord (AtmelGenericRecord * genericRecord, FILE * in)`

Reads an Atmel Generic record from an opened file.

##### Parameters

*genericRecord* A pointer to the Atmel Generic record structure that will store the read record.

*in* A file pointer to an opened file that can be read.

##### Returns

ATMEL\_GENERIC\_OK on success, otherwise one of the ATMEL\_GENERIC\_ERROR\_ error codes.

**Return values**

*ATMEL\_GENERIC\_OK* on success.

*ATMEL\_GENERIC\_ERROR\_INVALID\_ARGUMENTS* if genericRecord does not point to a valid AtmelGenericRecord structure or the file pointer is invalid.

*ATMEL\_GENERIC\_ERROR\_EOF* if end-of-file has been reached.

*ATMEL\_GENERIC\_ERROR\_FILE* if a file reading error has occurred.

*ATMEL\_GENERIC\_INVALID\_RECORD* if the record read is invalid (record did not match specifications).

**4.1.4.4 int Write\_AtmelGenericRecord (const AtmelGenericRecord genericRecord, FILE \* out)**

Writes an Atmel Generic record to an opened file. Note that the Atmel Generic record only supports 24-bit addresses, so only 24-bits of the address stored in the Atmel Generic record structure that genericRecord points to will be written.

**Parameters**

*genericRecord* The Atmel Generic record structure that will be written.

*out* A file pointer to an opened file that can be written to.

**Returns**

ATMEL\_GENERIC\_OK on success, otherwise one of the ATMEL\_GENERIC\_ERROR\_ error codes.

**Return values**

*ATMEL\_GENERIC\_OK* on success.

*ATMEL\_GENERIC\_ERROR\_INVALID\_ARGUMENTS* if the file pointer is invalid.

*ATMEL\_GENERIC\_ERROR\_FILE* if a file writing error has occurred.

## 4.2 ihex.h File Reference

Low-level utility functions to create, read, write, and print Intel HEX8 binary records.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
```

### Data Structures

- struct [\\_IHexRecord](#)

### Typedefs

- typedef struct [\\_IHexRecord](#) [IHexRecord](#)

### Enumerations

- enum [\\_IHexDefinitions](#) {  
    [IHEX\\_RECORD\\_BUFF\\_SIZE](#) = 768, [IHEX\\_COUNT\\_OFFSET](#) = 1, [IHEX\\_COUNT\\_LEN](#) = 2, [IHEX\\_ADDRESS\\_OFFSET](#) = 3,  
    [IHEX\\_ADDRESS\\_LEN](#) = 4, [IHEX\\_TYPE\\_OFFSET](#) = 7, [IHEX\\_TYPE\\_LEN](#) = 2, [IHEX\\_DATA\\_OFFSET](#) = 9,  
    [IHEX\\_CHECKSUM\\_LEN](#) = 2, [IHEX\\_MAX\\_DATA\\_LEN](#) = 512, [IHEX\\_ASCII\\_HEX\\_BYTE\\_LEN](#) = 2, [IHEX\\_START\\_CODE\\_OFFSET](#) = 0,  
    [IHEX\\_START\\_CODE](#) = ':' }  
• enum [IHexErrors](#) {  
    [IHEX\\_OK](#) = 0, [IHEX\\_ERROR\\_FILE](#) = -1, [IHEX\\_ERROR\\_EOF](#) = -2, [IHEX\\_ERROR\\_INVALID\\_RECORD](#) = -3,  
    [IHEX\\_ERROR\\_INVALID\\_ARGUMENTS](#) = -4, [IHEX\\_ERROR\\_NEWLINE](#) = -5 }  
• enum [IHexRecordTypes](#) {  
    [IHEX\\_TYPE\\_00](#) = 0, [IHEX\\_TYPE\\_01](#), [IHEX\\_TYPE\\_02](#), [IHEX\\_TYPE\\_03](#),  
    [IHEX\\_TYPE\\_04](#), [IHEX\\_TYPE\\_05](#) }

### Functions

- int [New\\_IHexRecord](#) (int type, uint16\_t address, uint8\_t data[], int dataLen, [IHexRecord](#) \*ihexRecord)
- int [Read\\_IHexRecord](#) ([IHexRecord](#) \*ihexRecord, FILE \*in)
- int [Write\\_IHexRecord](#) (const [IHexRecord](#) ihexRecord, FILE \*out)
- void [Print\\_IHexRecord](#) (const [IHexRecord](#) ihexRecord)
- uint8\_t [Checksum\\_IHexRecord](#) (const [IHexRecord](#) ihexRecord)

### 4.2.1 Detailed Description

Low-level utility functions to create, read, write, and print Intel HEX8 binary records.

#### Author

Vanya A. Sergeev <[vsergeev@gmail.com](mailto:vsergeev@gmail.com)>

#### Date

October 10 2009

#### Version

1.0.3

### 4.2.2 Typedef Documentation

#### 4.2.2.1 typedef struct \_IHexRecord IHexRecord

Alias "IHexRecord" for struct [\\_IHexRecord](#), done for convenience and clarity.

### 4.2.3 Enumeration Type Documentation

#### 4.2.3.1 enum IHexErrors

All possible error codes the Intel HEX8 record utility functions may return.

##### Enumerator:

***IHEX\_OK*** Error code for success or no error.

***IHEX\_ERROR\_FILE*** Error code for error while reading from or writing to a file. You may check `errno` for the exact error if this error code is encountered.

***IHEX\_ERROR\_EOF*** Error code for encountering end-of-file when reading from a file.

***IHEX\_ERROR\_INVALID\_RECORD*** Error code for error if an invalid record was read.

***IHEX\_ERROR\_INVALID\_ARGUMENTS*** Error code for error from invalid arguments passed to function.

***IHEX\_ERROR\_NEWLINE*** Error code for encountering a newline with no record when reading from a file.

#### 4.2.3.2 enum IHexRecordTypes

Intel HEX8 Record Types 00-05

##### Enumerator:

***IHEX\_TYPE\_00*** Data Record

***IHEX\_TYPE\_01*** End of File Record

***IHEX\_TYPE\_02*** Extended Segment Address Record

***IHEX\_TYPE\_03*** Start Segment Address Record

***IHEX\_TYPE\_04*** Extended Linear Address Record

***IHEX\_TYPE\_05*** Start Linear Address Record

## 4.2.4 Function Documentation

### 4.2.4.1 uint8\_t Checksum\_IHexRecord (const IHexRecord *ihexRecord*)

Calculates the checksum of an Intel HEX8 IHexRecord structure. See the Intel HEX8 specifications for more details on the checksum calculation.

#### Parameters

*ihexRecord* The Intel HEX8 record structure that the checksum will be calculated of.

#### Returns

The 8-bit checksum.

### 4.2.4.2 int New\_IHexRecord (int *type*, uint16\_t *address*, uint8\_t *data*[], int *dataLen*, IHexRecord \* *ihexRecord*)

Sets all of the record fields of an Intel HEX8 record structure.

#### Parameters

*type* The Intel HEX8 record type (integer value of 0 through 5).

*address* The 16-bit address of the data.

*data* The 8-bit array of data.

*dataLen* The size of the 8-bit data array.

*ihexRecord* A pointer to the target Intel HEX8 record structure where these fields will be set.

#### Returns

IHEX\_OK on success, otherwise one of the IHEX\_ERROR\_ error codes.

#### Return values

**IHEX\_OK** on success.

**IHEX\_ERROR\_INVALID\_ARGUMENTS** if *ihexRecord* does not point to a valid IHexRecord structure, or if the length of the 8-bit data array is out of range (less than zero or greater than the maximum data length allowed by record specifications, see [IHexRecord.data](#)).

### 4.2.4.3 void Print\_IHexRecord (const IHexRecord *ihexRecord*)

Prints the contents of an Intel HEX8 record structure to stdout. The record dump consists of the type, address, entire data array, and checksum fields of the record.

#### Parameters

*ihexRecord* The Intel HEX8 record structure that will be printed to stdout.

#### Returns

Always returns IHEX\_OK (success).

#### Return values

**IHEX\_OK** on success.

#### 4.2.4.4 int Read\_IHexRecord (IHexRecord \* *ihexRecord*, FILE \* *in*)

Reads an Intel HEX8 record from an opened file.

##### Parameters

*ihexRecord* A pointer to the Intel HEX8 record structure that will store the read record.

*in* A file pointer to an opened file that can be read.

##### Returns

IHEX\_OK on success, otherwise one of the IHEX\_ERROR\_ error codes.

##### Return values

**IHEX\_OK** on success.

**IHEX\_ERROR\_INVALID\_ARGUMENTS** if *ihexRecord* does not point to a valid IHexRecord structure or the file pointer "*in*" is invalid.

**IHEX\_ERROR\_EOF** if end-of-file has been reached.

**IHEX\_ERROR\_FILE** if a file reading error has occurred.

**IHEX\_INVALID\_RECORD** if the record read is invalid (record did not match specifications or record checksum was invalid).

#### 4.2.4.5 int Write\_IHexRecord (const IHexRecord *ihexRecord*, FILE \* *out*)

Writes an Intel HEX8 record to an opened file.

##### Parameters

*ihexRecord* The Intel HEX8 record structure that will be written.

*out* A file pointer to an opened file that can be written to.

##### Returns

IHEX\_OK on success, otherwise one of the IHEX\_ERROR\_ error codes.

##### Return values

**IHEX\_OK** on success.

**IHEX\_ERROR\_INVALID\_ARGUMENTS** if the file pointer "*out*" is invalid.

**IHEX\_ERROR\_INVALID\_RECORD** if the record's data length is out of range (greater than the maximum data length allowed by record specifications, see [IHexRecord.data](#)).

**IHEX\_ERROR\_FILE** if a file writing error has occurred.

## 4.3 srecord.h File Reference

Low-level utility functions to create, read, write, and print Motorola S-Record binary records.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
```

### Data Structures

- struct [\\_SRecord](#)

### Typedefs

- typedef struct [\\_SRecord](#) [SRecord](#)

### Enumerations

- enum [\\_SRecordDefinitions](#) {  
    **SRECORD\_RECORD\_BUFF\_SIZE** = 768, **SRECORD\_TYPE\_OFFSET** = 1, **SRECORD\_TYPE\_LEN** = 1, **SRECORD\_COUNT\_OFFSET** = 2,  
    **SRECORD\_COUNT\_LEN** = 2, **SRECORD\_ADDRESS\_OFFSET** = 4, **SRECORD\_CHECKSUM\_LEN** = 2, **SRECORD\_MAX\_DATA\_LEN** = 64,  
    **SRECORD\_MAX\_ADDRESS\_LEN** = 8, **SRECORD\_ASCII\_HEX\_BYTE\_LEN** = 2,  
    **SRECORD\_START\_CODE\_OFFSET** = 0, **SRECORD\_START\_CODE** = 'S' }  
• enum [SRecordErrors](#) {  
    **SRECORD\_OK** = 0, **SRECORD\_ERROR\_FILE** = -1, **SRECORD\_ERROR\_EOF** = -2,  
    **SRECORD\_ERROR\_INVALID\_RECORD** = -3,  
    **SRECORD\_ERROR\_INVALID\_ARGUMENTS** = -4, **SRECORD\_ERROR\_NEWLINE** = -5 }  
• enum [SRecordTypes](#) {  
    **SRECORD\_TYPE\_S0** = 0, **SRECORD\_TYPE\_S1**, **SRECORD\_TYPE\_S2**, **SRECORD\_TYPE\_S3**,  
    **SRECORD\_TYPE\_S4**, **SRECORD\_TYPE\_S5**, **SRECORD\_TYPE\_S6**, **SRECORD\_TYPE\_S7**,  
    **SRECORD\_TYPE\_S8**, **SRECORD\_TYPE\_S9** }

### Functions

- int [New\\_SRecord](#) (int type, uint32\_t address, uint8\_t data[ ], int dataLen, [SRecord](#) \*srec)
- int [Read\\_SRecord](#) ([SRecord](#) \*srec, FILE \*in)
- int [Write\\_SRecord](#) (const [SRecord](#) srec, FILE \*out)
- void [Print\\_SRecord](#) (const [SRecord](#) srec)
- uint8\_t [Checksum\\_SRecord](#) (const [SRecord](#) srec)

### 4.3.1 Detailed Description

Low-level utility functions to create, read, write, and print Motorola S-Record binary records.

**Author**

Vanya A. Sergeev <[vsergeev@gmail.com](mailto:vsergeev@gmail.com)>

**Date**

October 10 2009

**Version**

1.0.3

### 4.3.2 Typedef Documentation

#### 4.3.2.1 typedef struct `_SRecord` `SRecord`

Alias "SRecord" for struct `_SRecord`, done for convenience and clarity.

### 4.3.3 Enumeration Type Documentation

#### 4.3.3.1 enum `SRecordErrors`

All possible error codes the Motorola S-Record utility functions may return.

**Enumerator:**

**`SRECORD_OK`** Error code for success or no error.

**`SRECORD_ERROR_FILE`** Error code for error while reading from or writing to a file. You may check `errno` for the exact error if this error code is encountered.

**`SRECORD_ERROR_EOF`** Error code for encountering end-of-file when reading from a file.

**`SRECORD_ERROR_INVALID_RECORD`** Error code for error if an invalid record was read.

**`SRECORD_ERROR_INVALID_ARGUMENTS`** Error code for error from invalid arguments passed to function.

**`SRECORD_ERROR_NEWLINE`** Error code for encountering a newline with no record when reading from a file.

#### 4.3.3.2 enum `SRecordTypes`

Motorola S-Record Types S0-S9

**Enumerator:**

**`SRECORD_TYPE_S0`** Header record, although there is an official format it is often made proprietary by third-parties. 16-bit address normally set to 0x0000 and header information is stored in the data field. This record is unnecessary and commonly not used.

**`SRECORD_TYPE_S1`** Data record with 16-bit address

**`SRECORD_TYPE_S2`** Data record with 24-bit address



- SRECORD\_TYPE\_S3*** Data record with 32-bit address
- SRECORD\_TYPE\_S4*** Extension by LSI Logic, Inc. See their specification for more details.
- SRECORD\_TYPE\_S5*** 16-bit address field that contains the number of S1, S2, and S3 (all data) records transmitted. No data field.
- SRECORD\_TYPE\_S6*** 24-bit address field that contains the number of S1, S2, and S3 (all data) records transmitted. No data field.
- SRECORD\_TYPE\_S7*** Termination record for S3 data records. 32-bit address field contains address of the entry point after the S-Record file has been processed. No data field.
- SRECORD\_TYPE\_S8*** Termination record for S2 data records. 24-bit address field contains address of the entry point after the S-Record file has been processed. No data field.
- SRECORD\_TYPE\_S9*** Termination record for S1 data records. 16-bit address field contains address of the entry point after the S-Record file has been processed. No data field.

### 4.3.4 Function Documentation

#### 4.3.4.1 `uint8_t` **Checksum\_SRecord** (`const SRecord srec`)

Calculates the checksum of a Motorola S-Record SRecord structure. See the Motorola S-Record specifications for more details on the checksum calculation.

##### Parameters

***srec*** The Motorola S-Record structure that the checksum will be calculated of.

##### Returns

The 8-bit checksum.

#### 4.3.4.2 `int` **New\_SRecord** (`int type`, `uint32_t address`, `uint8_t data[]`, `int dataLen`, `SRecord * srec`)

Sets all of the record fields of a Motorola S-Record structure.

##### Parameters

***type*** The Motorola S-Record type (integer value of 0 through 9).

***address*** The 32-bit address of the data. The actual size of the address (16-,24-, or 32-bits) when written to a file depends on the S-Record type.

***data*** The 8-bit array of data.

***dataLen*** The size of the 8-bit data array.

***srec*** A pointer to the target Motorola S-Record structure where these fields will be set.

##### Returns

***SRECORD\_OK*** on success, otherwise one of the ***SRECORD\_ERROR\_*** error codes.

##### Return values

***SRECORD\_OK*** on success.

***SRECORD\_ERROR\_INVALID\_ARGUMENTS*** if *srec* does not point to a valid SRecord structure, or if the length of the 8-bit data array is out of range (less than zero or greater than the maximum data length allowed by record specifications, see [SRecord.data](#)).

#### 4.3.4.3 void Print\_SRecord (const SRecord *srec*)

Prints the contents of a Motorola S-Record structure to stdout. The record dump consists of the type, address, entire data array, and checksum fields of the record.

##### Parameters

*srec* The Motorola S-Record structure that will be printed to stdout.

##### Returns

Always returns SRECORD\_OK (success).

##### Return values

**SRECORD\_OK** on success.

#### 4.3.4.4 int Read\_SRecord (SRecord \* *srec*, FILE \* *in*)

Reads a Motorola S-Record record from an opened file.

##### Parameters

*srec* A pointer to the Motorola S-Record structure that will store the read record.

*in* A file pointer to an opened file that can be read.

##### Returns

SRECORD\_OK on success, otherwise one of the SRECORD\_ERROR\_ error codes.

##### Return values

**SRECORD\_OK** on success.

**SRECORD\_ERROR\_INVALID\_ARGUMENTS** if *srec* does not point to a valid SRecord structure or the file pointer "*in*" is invalid.

**SRECORD\_ERROR\_EOF** if end-of-file has been reached.

**SRECORD\_ERROR\_FILE** if a file reading error has occurred.

**SRECORD\_INVALID\_RECORD** if the record read is invalid (record did not match specifications or record checksum was invalid).

#### 4.3.4.5 int Write\_SRecord (const SRecord *srec*, FILE \* *out*)

Writes a Motorola S-Record to an opened file.

##### Parameters

*srec* The Motorola S-Record structure that will be written.

*out* A file pointer to an opened file that can be written to.

##### Returns

SRECORD\_OK on success, otherwise one of the SRECORD\_ERROR\_ error codes.

**Return values**

***SRECORD\_OK*** on success.

***SRECORD\_ERROR\_INVALID\_ARGUMENTS*** if the file pointer "out" is invalid.

***SRECORD\_ERROR\_INVALID\_RECORD*** if the record's data length (the [SRecord.dataLen](#) variable of the record) is out of range (greater than the maximum data length allowed by record specifications, see [SRecord.data](#)).

***SRECORD\_ERROR\_FILE*** if a file writing error has occurred.

# Index

- [\\_AtmelGenericRecord](#), [5](#)
    - [address](#), [5](#)
    - [data](#), [5](#)
  - [\\_IHexRecord](#), [6](#)
    - [address](#), [6](#)
    - [checksum](#), [6](#)
    - [data](#), [6](#)
    - [dataLen](#), [6](#)
    - [type](#), [6](#)
  - [\\_SRecord](#), [7](#)
    - [address](#), [7](#)
    - [checksum](#), [7](#)
    - [data](#), [7](#)
    - [dataLen](#), [7](#)
    - [type](#), [7](#)
- [address](#)
  - [\\_AtmelGenericRecord](#), [5](#)
  - [\\_IHexRecord](#), [6](#)
  - [\\_SRecord](#), [7](#)
- [atmel\\_generic.h](#)
  - [ATMEL\\_GENERIC\\_ERROR\\_EOF](#), [10](#)
  - [ATMEL\\_GENERIC\\_ERROR\\_FILE](#), [10](#)
  - [ATMEL\\_GENERIC\\_ERROR\\_INVALID\\_ARGUMENTS](#), [10](#)
  - [ATMEL\\_GENERIC\\_ERROR\\_INVALID\\_RECORD](#), [10](#)
  - [ATMEL\\_GENERIC\\_ERROR\\_NEWLINE](#), [10](#)
  - [ATMEL\\_GENERIC\\_OK](#), [10](#)
- [ATMEL\\_GENERIC\\_ERROR\\_EOF](#)
  - [atmel\\_generic.h](#), [10](#)
- [ATMEL\\_GENERIC\\_ERROR\\_FILE](#)
  - [atmel\\_generic.h](#), [10](#)
- [ATMEL\\_GENERIC\\_ERROR\\_INVALID\\_ARGUMENTS](#)
  - [atmel\\_generic.h](#), [10](#)
- [ATMEL\\_GENERIC\\_ERROR\\_INVALID\\_RECORD](#)
  - [atmel\\_generic.h](#), [10](#)
- [ATMEL\\_GENERIC\\_ERROR\\_NEWLINE](#)
  - [atmel\\_generic.h](#), [10](#)
- [ATMEL\\_GENERIC\\_OK](#)
  - [atmel\\_generic.h](#), [10](#)
- [atmel\\_generic.h](#), [9](#)
  - [AtmelGenericErrors](#), [10](#)
  - [AtmelGenericRecord](#), [10](#)
  - [New\\_AtmelGenericRecord](#), [11](#)
  - [Print\\_AtmelGenericRecord](#), [11](#)
  - [Read\\_AtmelGenericRecord](#), [11](#)
  - [Write\\_AtmelGenericRecord](#), [12](#)
- [AtmelGenericErrors](#)
  - [atmel\\_generic.h](#), [10](#)
- [AtmelGenericRecord](#)
  - [atmel\\_generic.h](#), [10](#)
- [checksum](#)
  - [\\_IHexRecord](#), [6](#)
  - [\\_SRecord](#), [7](#)
- [Checksum\\_IHexRecord](#)
  - [ihex.h](#), [15](#)
- [Checksum\\_SRecord](#)
  - [srecord.h](#), [19](#)
- [data](#)
  - [\\_AtmelGenericRecord](#), [5](#)
  - [\\_IHexRecord](#), [6](#)
  - [\\_SRecord](#), [7](#)
- [dataLen](#)
  - [\\_IHexRecord](#), [6](#)
  - [\\_SRecord](#), [7](#)
- [ihex.h](#), [13](#)
  - [Checksum\\_IHexRecord](#), [15](#)
  - [IHEX\\_ERROR\\_EOF](#), [14](#)
  - [IHEX\\_ERROR\\_FILE](#), [14](#)
  - [IHEX\\_ERROR\\_INVALID\\_ARGUMENTS](#), [14](#)
  - [IHEX\\_ERROR\\_INVALID\\_RECORD](#), [14](#)
  - [IHEX\\_ERROR\\_NEWLINE](#), [14](#)
  - [IHEX\\_OK](#), [14](#)
  - [IHEX\\_TYPE\\_00](#), [14](#)
  - [IHEX\\_TYPE\\_01](#), [14](#)
  - [IHEX\\_TYPE\\_02](#), [14](#)
  - [IHEX\\_TYPE\\_03](#), [14](#)
  - [IHEX\\_TYPE\\_04](#), [14](#)
  - [IHEX\\_TYPE\\_05](#), [14](#)
  - [IHexErrors](#), [14](#)
  - [IHexRecord](#), [14](#)
  - [IHexRecordTypes](#), [14](#)
  - [New\\_IHexRecord](#), [15](#)

- Print\_IHexRecord, [15](#)
- Read\_IHexRecord, [15](#)
- Write\_IHexRecord, [16](#)
- IHEX\_ERROR\_EOF
  - ihex.h, [14](#)
- IHEX\_ERROR\_FILE
  - ihex.h, [14](#)
- IHEX\_ERROR\_INVALID\_ARGUMENTS
  - ihex.h, [14](#)
- IHEX\_ERROR\_INVALID\_RECORD
  - ihex.h, [14](#)
- IHEX\_ERROR\_NEWLINE
  - ihex.h, [14](#)
- IHEX\_OK
  - ihex.h, [14](#)
- IHEX\_TYPE\_00
  - ihex.h, [14](#)
- IHEX\_TYPE\_01
  - ihex.h, [14](#)
- IHEX\_TYPE\_02
  - ihex.h, [14](#)
- IHEX\_TYPE\_03
  - ihex.h, [14](#)
- IHEX\_TYPE\_04
  - ihex.h, [14](#)
- IHEX\_TYPE\_05
  - ihex.h, [14](#)
- IHexErrors
  - ihex.h, [14](#)
- IHexRecord
  - ihex.h, [14](#)
- IHexRecordTypes
  - ihex.h, [14](#)
- New\_AtmelGenericRecord
  - atmel\_generic.h, [11](#)
- New\_IHexRecord
  - ihex.h, [15](#)
- New\_SRecord
  - srecord.h, [19](#)
- Print\_AtmelGenericRecord
  - atmel\_generic.h, [11](#)
- Print\_IHexRecord
  - ihex.h, [15](#)
- Print\_SRecord
  - srecord.h, [19](#)
- Read\_AtmelGenericRecord
  - atmel\_generic.h, [11](#)
- Read\_IHexRecord
  - ihex.h, [15](#)
- Read\_SRecord
  - srecord.h, [20](#)
- SRecord
  - srecord.h, [18](#)
- srecord.h, [17](#)
  - Checksum\_SRecord, [19](#)
  - New\_SRecord, [19](#)
  - Print\_SRecord, [19](#)
  - Read\_SRecord, [20](#)
  - SRecord, [18](#)
  - SRECORD\_ERROR\_EOF, [18](#)
  - SRECORD\_ERROR\_FILE, [18](#)
  - SRECORD\_ERROR\_INVALID\_ARGUMENTS, [18](#)
  - SRECORD\_ERROR\_INVALID\_RECORD, [18](#)
  - SRECORD\_ERROR\_NEWLINE, [18](#)
  - SRECORD\_OK, [18](#)
  - SRECORD\_TYPE\_S0, [18](#)
  - SRECORD\_TYPE\_S1, [18](#)
  - SRECORD\_TYPE\_S2, [18](#)
  - SRECORD\_TYPE\_S3, [18](#)
  - SRECORD\_TYPE\_S4, [19](#)
  - SRECORD\_TYPE\_S5, [19](#)
  - SRECORD\_TYPE\_S6, [19](#)
  - SRECORD\_TYPE\_S7, [19](#)
  - SRECORD\_TYPE\_S8, [19](#)
  - SRECORD\_TYPE\_S9, [19](#)
  - SRecordErrors, [18](#)
  - SRecordTypes, [18](#)
  - Write\_SRecord, [20](#)
- SRECORD\_ERROR\_EOF
  - srecord.h, [18](#)
- SRECORD\_ERROR\_FILE
  - srecord.h, [18](#)
- SRECORD\_ERROR\_INVALID\_ARGUMENTS
  - srecord.h, [18](#)
- SRECORD\_ERROR\_INVALID\_RECORD
  - srecord.h, [18](#)
- SRECORD\_ERROR\_NEWLINE
  - srecord.h, [18](#)
- SRECORD\_OK
  - srecord.h, [18](#)
- SRECORD\_TYPE\_S0
  - srecord.h, [18](#)
- SRECORD\_TYPE\_S1
  - srecord.h, [18](#)
- SRECORD\_TYPE\_S2
  - srecord.h, [18](#)
- SRECORD\_TYPE\_S3
  - srecord.h, [18](#)
- SRECORD\_TYPE\_S4
  - srecord.h, [19](#)
- SRECORD\_TYPE\_S5
  - srecord.h, [19](#)
- SRECORD\_TYPE\_S6
  - srecord.h, [19](#)

- srecord.h, [19](#)
- SRECORD\_TYPE\_S7
  - srecord.h, [19](#)
- SRECORD\_TYPE\_S8
  - srecord.h, [19](#)
- SRECORD\_TYPE\_S9
  - srecord.h, [19](#)
- SRecordErrors
  - srecord.h, [18](#)
- SRecordTypes
  - srecord.h, [18](#)
- type
  - \_IHexRecord, [6](#)
  - \_SRecord, [7](#)
- Write\_AtmelGenericRecord
  - atmel\_generic.h, [12](#)
- Write\_IHexRecord
  - ihex.h, [16](#)
- Write\_SRecord
  - srecord.h, [20](#)